Optimization for Active Learning-based Interactive Database Exploration

Enhui Huang[†], Liping Peng[*], Luciano Di Palma[†], Ahmed Abdelkafi[†], Anna Liu[*], Yanlei Diao[*,†]
[†]École Polytechnique, France; [*]University of Massachusetts Amherst, USA
[†]{enhui.huang, luciano.di-palma, ahmed.abdelkafi}@polytechnique.edu,
[*]{lppeng, yanlei}@cs.umass.edu, anna@math.umass.edu

ABSTRACT

There is an increasing gap between fast growth of data and limited human ability to comprehend data. Consequently, there has been a growing demand of data management tools that can bridge this gap and help the user retrieve high-value content from data more effectively. In this work, we aim to build interactive data exploration as a new database service, using an approach called "explore-by-example". In particular, we cast the explore-by-example problem in a principled "active learning" framework, and bring the properties of important classes of database queries to bear on the design of new algorithms and optimizations for active learning-based database exploration. These new techniques allow the database system to overcome a fundamental limitation of traditional active learning, i.e., the slow convergence problem. Evaluation results using real-world datasets and user interest patterns show that our new system significantly outperforms state-of-the-art active learning techniques and data exploration systems in accuracy while achieving desired efficiency for interactive performance.

PVLDB Reference Format:
Enhui Huang, Liping Peng, Luciano Di Palma, Ahmed Abdelkafi, Anna Liu, Yanlei Diao. Optimization for Active Learning-based Interactive Database Exploration. *PVLDB*, 12(1): 71-84, 2018.
DOI: https://doi.org/10.14778/3275536.3275542

1. INTRODUCTION

Today data is being generated at an unprecedented rate. However, the human ability to comprehend data remains as limited as before. Consequently, there has been a growing demand of data management tools that can bridge the gap between the data growth and limited human ability, and help retrieve high-value content from data more effectively.

To respond to such needs, we build a new database service for interactive exploration in a framework called "*explore-by-example*" [12, 13]. In this framework, the database content is considered as a set of tuples, and the user is interested in some of them but not all. In the data exploration process, the system allows the user to interactively label tuples as "interesting" or "not interesting", so that it can construct an increasingly-more-accurate model of the user interest. Eventually, the model is turned into a *user interest query*[1] that will retrieve all relevant tuples from the database.

In this work, we consider several target applications. First, when a scientist comes to explore a large sky survey database such as SDSS [39], she may not be able to express her data interest precisely. Instead, she may prefer to navigate through a region of the sky, see a few examples of sky objects, provide yes or no feedback, and ask the system to find all other (potentially many more) relevant sky objects from the database. Second, consider many web applications backed by a large database, such as E-commerce websites and housing websites, which provide a simple search interface but leave the job of filtering through a long list of returned objects to the user. The new database service in the explore-by-example framework will provide these applications with a new way to interact with the user and, more importantly, help the user filter through numerous objects more efficiently.

Our approach to building an explore-by-example system is to cast it in an "*active learning*" framework: We treat the modeling of the user interest as a *classification* problem where all the user labeled examples thus far are used to train a classification model. Then active learning [5, 36, 41] decides how to choose a new example, from the unlabeled database, for the user to label next so that the system can learn the user interest efficiently. The active learning based explore-by-example approach offers potential benefits over alternative methods such as faceted search [26, 34, 35] and semantic windows [25] for several reasons.

First, the user interest may include varying degrees of complexity, or the user does not have prior knowledge about the complexity and hence expects the system to learn a model as complex as necessary for her interest. More specifically, if the user knows the relevant attributes and just wants to set the appropriate value for each attribute, an interactive GUI or faceted search [26, 34, 35] may be sufficient. However, the user interest may involve more complex constraints, e.g., "$(\frac{rowc-a_1}{b_1})^2+(\frac{rowc-a_2}{b_2})^2 < c$", and "$x+2.5*\log_{10}(y) < 23.3$" from the SDSS example query set [38], or "$length*width > c$" as seen in our car database. If the user knows the function shape and constants in advance, some of the above examples (e.g., the ellipse pattern) can be supported by semantic

[1]In our work we use the term, *user interest query*, to refer to the final query that represents the user interest, while the term, *user interest model*, can refer to an immediate model before it converges to the true user interest.

Proceedings of the VLDB Endowment, Vol. 12, No. 1
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3275536.3275542

windows [25] as pre-defined patterns. However, if the user does not have such prior knowledge, explore-by-example may work regardless of how complex the predicates are, and which functions and constants are used in the predicates.

Second, increased dimensionality makes it harder for semantic windows to scale. For example, when the interest of the SDSS user involves both the ellipse and log patterns above, it will be more difficult for both the system and the user to handle multiple patterns for data exploration (even if such patterns can be predefined). In contrast, as the dimensionality increases, explore-by-example can keep the same user interface for data exploration and handles increased complexity via its learning algorithm "behind the scenes".

While prior work on explore-by-example has used active learning [13], a main issue is the large number of labeled examples needed to achieve high accuracy. For example, 300-500 labeled examples are needed to reach 80% accuracy [13], which is undesirable in many applications. This problem, referred to as *slow convergence*, is exacerbated when the user interest covers only a small fraction of the database (i.e., low selectivity) or the number of the attributes chosen for exploration is large (i.e., high dimensionality).

In this work, we take a new approach to active learning-based database exploration. Instead of improving active learning in isolation from the database, we treat it as an internal module of the database system and ask the question: *what query and data properties from the database can we leverage to address the slow convergence problem?* Based on the common properties that we observed in query traces from the Sloan Digital Sky Survey [38] and a car database (described more in Section 6), we can indeed design new techniques that overcome or alleviate the slow convergence problem. Some of the key query properties include:

Subspatial Convexity: Consider the database attributes as dimensions of a data space $\mathcal{D}$ and map the tuples to the space based on the values of their attributes. Then all the tuples that match the user interest form the positive region in $\mathcal{D}$; others form the negative region. We observe that in some lower-dimensional subspaces of $\mathcal{D}$, the projected positive or negative region is a convex object. For example, the SDSS query trace [38] includes 116 predicates, among which 107 predicates define a convex positive region in the subspace formed by the attributes used in the predicate, and 3 predicates define a convex negative region in their subspaces. For the car database, the 70 predicates defined on numerical attributes all form a convex positive region. Figure 2 shows a range of predicates from these two databases and their positive and negative regions.

Conjunctivity: Conjunctive queries are a major class of database queries that have been used in numerous applications. For data exploration, we are interested in the "conjunctive" property of the set of predicates that characterize the user interest. Among 45 queries provided by SDSS [38], 40 queries are conjunctive queries. For the car database, all user queries use the conjunctive property.

In this paper, we bring the subspatial convexity and conjunctive properties of database queries, treated as true (yet unknown) user interest queries, to bear on the design of new algorithms and optimizations for active learning-based database exploration. These techniques allow the database system to overcome a fundamental limitation of traditional active learning, i.e., the slow convergence problem when data exploration is performed with high dimensionality and low selectivity of the user interest query. More specifically, our paper makes the following contributions.

1. Dual-Space Model (Section 3): By leveraging the subspatial convex property, we propose a new "dual-space model" (DSM) that builds not only a classification model, $F_{\mathcal{V}}$, from labeled examples, but also a polytope model of the data space, $F_{\mathcal{D}}$. On one hand, active learning theory improves $F_{\mathcal{V}}$ by choosing the next example that enables reduction of the version space $\mathcal{V}$ (the space of all classification models consistent with labeled data). On the other hand, our polytope model offers a more direct description of the data space $\mathcal{D}$ including the areas known to be positive, areas known to be negative, and areas with unknown labels. We use both models to predict unlabeled examples and choose the best example to label next. In addition, DSM allows us to prove exact and approximate lower bounds on the model accuracy in terms of F1-score. While active learning theory offers provable bounds on classification errors [7, 14, 18–20], it treats positive and negative classes equally. Given the low selectivity of user interest queries, e.g., 1%, a classifier that assigns all tuples to the negative class has a low error rate of 1%, but fails to return any relevant tuples. Hence, we choose to bound F1-score as it emphasizes accuracy for the positive class, i.e., the relevant tuples returned to the user.

2. High-dimensional Exploration: When the user interest involves a large number of attributes, we employ two approaches to reducing dimensionality in data exploration.

(a) *Factorization* (Section 4): By leveraging the conjunctive and subspatial convexity properties of user interest queries, we factorize a high-dimensional data space into low-dimensional subspaces, in some of which the projections of user positive or negative regions are convex. Our dual-space model with factorization, DSM$_F$, runs the polytope model in each subspace where the convex property holds. We formally define the class of queries that DSM$_F$ supports, the decision function it utilizes, and prove that it achieves a better lower bound of F1-score than DSM without factorization.

(b) *Online feature selection* (Section 5): The user may start exploration with more attributes than those needed in the final model. To remove irrelevant attributes, we propose an online feature selection method that adaptively selects the top-k relevant attributes and leverages the convex property (if present) for optimization.

3. Evaluation (Section 6): We evaluated our system using two real datasets. The SDSS dataset [39] includes 190M tuples, for which the user interests are selective and their decision boundaries present varied complexity for detection. Without knowing these queries in advance, DSM$_F$ can achieve F1-score of 95% within 10 labeled examples if the decision boundary B falls in a sparse region, and otherwise requires up to 100 labeled examples for 2D queries and 160-240 examples for 4D-6D queries while maintaining per-iteration time within 1-2 seconds. For these queries, our results show that DSM$_F$ significantly outperforms learning methods including Active Learning (AL) [5,15] and Active Search [16], as well as recent explore-by-example systems, Aide [12, 13] and LifeJoin [9].

Our user study using a 5622-tuple car database validates the subspatial convex property and the conjunctive property of user interest queries. It also shows the benefits of DSM$_F$ (a median of 10 labeled examples to reach high accuracy) over manual exploration (where the user wrote 12 queries and reviewed 98 tuples as the median), as well as over AL, even in the presence of noisy user labels.

2. BACKGROUND

In this section, we review our design of an explore-by-example system and present background on active learning.

2.1 System Overview

Our data exploration system is depicted in Figure 1. The main concepts and modules are described as follows.

Data space. When a user comes to explore a database, she is presented with the database schema for browsing. Based on her best understanding of the (implicit) exploration goal, she may choose a set of attributes, $\{A_i\}$, $i = 1, \ldots, d$, from a table for consideration. These attributes form a superset of the relevant attributes that will eventually be discovered by the system. Let us consider the projection of the underlying table to $\{A_i\}$, and pivot the projected table such that each A_i becomes a dimension and the projected tuples are mapped to points in this d-dimensional space – the resulting space is called a *data space* where the user exploration will take place.

Initial examples. To bootstrap data exploration, the user is asked to give a positive example and a negative example. If she does not have such examples, the system can run initial sampling [12, 29] over the data space to help her find such examples. Since the initial sampling problem has been studied before, our work in this paper focuses on data exploration after such initial examples are identified.

Iterative learning and exploration. The iterative exploration starts with a given positive example set and a negative example set, which form the labeled dataset. In each iteration, the labeled dataset is used to train a user interest model, which is fast due to the small size of training data. Before the model reaches convergence or a user-specified accuracy level, it is used next to explore the data space and retrieve a new example for display. In the next iteration, the user labels this example as positive or negative – such feedback can be collected explicitly through a graphical interface [11], or implicitly based on the user behavior.[2] The newly labeled example is added to the labeled dataset, and the above process repeats.

Convergence and final retrieval. At each iteration, our system assesses the current model to decide whether more iterations are needed. The process is terminated when the model accuracy has reached a user-defined threshold. At this point, the model for the positive class is translated to a query which will retrieve from the database all the tuples classified as relevant. To provide better interpretability, our system can visualize the final (nonparametric) model and fit a corresponding parametric model, the form of which can be suggested by the user after seeing the visualization.

2.2 Active Learning for Data Exploration

The problem of dynamically seeking the next example for labeling from a large database of unlabeled tuples is closely related to active learning. The recent results on active learning are surveyed in [36]. Below, we summarize those results relevant to our work.

Pool-Based Sampling. Many real-world problems fit the following scenario: there is a small set of labeled data $\mathcal{L}$ and a large pool of unlabeled data $\mathcal{U}$ available. In active learning, an example is chosen from the pool in a greedy fashion,

[2]Methods for collecting user feedback are in the purview of human-computer interaction and are beyond the scope of this paper.

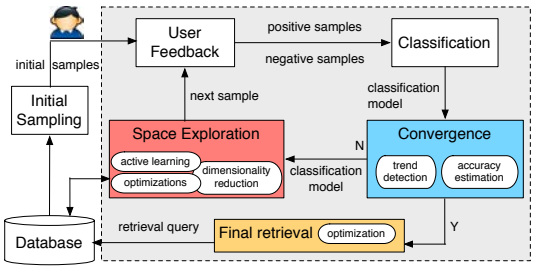


Figure 1: System architecture for explore by example.

according to a utility measure used to evaluate all instances in the pool (or, if $\mathcal{U}$ is large, a subsample thereof). In our setting of database exploration, the labeled data $\mathcal{L}$ is what the user has provided thus far. The pool $\mathcal{U}$ is a subsample of size m of the unlabeled part of the database. The utility measure depends on the classifier in use, as discussed below.

Classification Model. Previous explore-by-example systems [12, 13] used decision trees to build a classification model. It works well if the user interest pattern is a hyper-rectangle in the data space, whereas real-world applications may use more complex predicates. To support higher complexity, our system uses more powerful classifiers such as Support Vector Machines (SVM) or Gradient Boosting. The new techniques proposed in our work do **not** depend on the specific classifier; they can work with most existing classifiers. But the implementation of active learning does depend on the classifier in use. For ease of composition, in this paper we use SVM as an example classifier.

Uncertainty Sampling and Version Space Search. A common form of utility measure over the unlabeled pool $\mathcal{U}$ characterizes the degree of *uncertainty* that the current model experiences for classifying each data example in $\mathcal{U}$. Under uncertainty sampling, the example that leads to the highest degree of uncertainty in classification is chosen as the example for labeling. If the classifier is SVM-based, the uncertainty can be measured by the *distance* of each example from the decision boundary of the current SVM [5, 15]. A formal description of uncertainty sampling is through the notion of *version space*, which is the space of all configurations of the classification model consistent with labeled data. Uncertainty sampling is a (rough) approximation of an optimal algorithm that bisects the version space with each selected example [41]. For the above reason, we call active learning algorithms *version space-based* because they are designed to reduce the version space.

3. DUAL-SPACE MODEL

For interactive data exploration, active learning algorithms often require a large number of user-labeled examples to reach high accuracy, known as the slow convergence problem. One reason for slow convergence is the limitation of uncertainty sampling itself: it may exert a lot of effort searching in sparse, noisy, or irrelevant regions of the input space [36].

With the goal to provide a service for database exploration, our work takes a new, a database centric approach to tackle the slow convergence problem. We observe from existing query traces that in some lower-dimensional subspaces, the projected positive or negative region of user interest query is often a convex object. In this section, we utilize such subspatial convexity and introduce a dual-space (data and

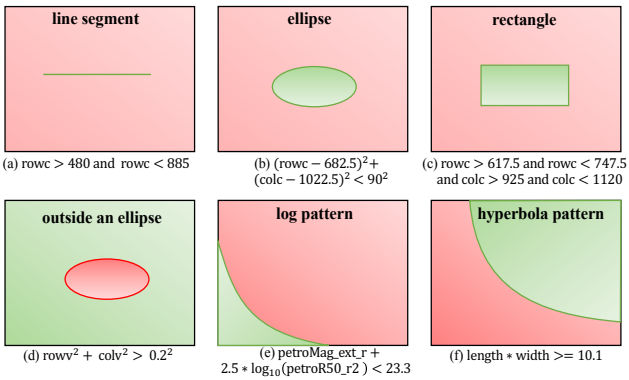
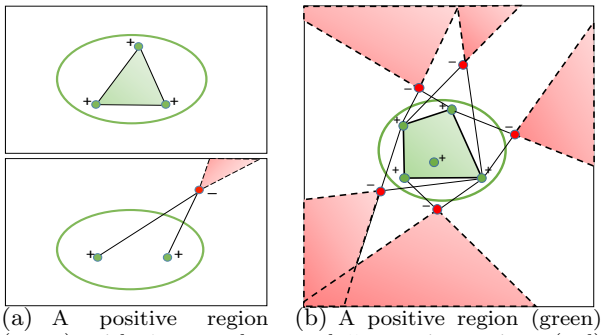


Figure 2: Positive (green) and negative (red) regions of 6 example predicates



(a) A positive region (green) with 3 examples, and a negative region (red) with 1 negative and 2 positive examples

(b) A positive region (green) and 5 negative regions (red) built from 5 positive examples and 5 negative examples.

Figure 3: Positive and negative regions in the polytope model.

version space) model, which enables improved accuracy and provable lower bounds on the model accuracy. We begin with the simple case that the convex property holds for the query over the entire data space $\mathcal{D}$, without factorization, and defer the extension to subspaces to Section 4. We call this class of queries "*convex pattern queries*", denoted as $\mathbf{Q}_c \equiv \mathbf{Q}_c^+ \cup \mathbf{Q}_c^-$. Figure 2 gives examples in both classes.

3.1 A Polytope Model in Data Space

The key idea behind our *data space* model is that at each iteration we use all available labeled examples to build a partitioning of the data space. It divides the data space into the *positive region* (any point inside which is known to be positive), the *negative region* (any point inside which is known to be negative) and the *uncertain region*. As more examples are labeled, we have more knowledge about the uncertain region, so part of it will be converted to either the positive or the negative region in later iterations. Eventually, with enough training data, the uncertain region becomes empty, and the positive region converges to the query region.

For simplicity, we begin with queries whose positive region is convex ($\mathbf{Q}_c^+$). When the query region Q is convex, any point on the line segment between two points $\boldsymbol{x}_1 \in Q$ and $\boldsymbol{x}_2 \in Q$ is also in Q. Then we have the following definition.

Definition 3.1 (Positive Region) *Denote the examples that have been labeled as "positive" as* $L^+ = \{e_i^+ \mid i = 1, \ldots, n^+\}$. *The convex hull of* L^+ *is known to be the smallest convex set that contains* L^+ *[27] and is called the positive region, denoted as* R^+.

It is known that the convex hull of a finite number of points is a *convex polytope* [17]. For example, the green triangle in Fig. 3(a) and the green tetragon in Fig. 3(b) are the positive regions formed by three and four positive examples, respectively, which are an approximation of the query region marked by the green ellipse. We can prove the following property of the positive region for convex queries, assuming that user labels are consistent with her interest:

Proposition 3.1 *All points in the positive region* R^+ *are positive.*

All proofs in this paper are left to Appendix A of our technical report [22] due to space constraints.

Definition 3.2 (Negative Region) *For a negative example* $\boldsymbol{e}_i^-$, *we can define a corresponding negative region* R_i^- *such that the line segment connecting any point* $\boldsymbol{x} \in R_i^-$ *and* $\boldsymbol{e}_i^-$ *does not overlap with the positive region* R^+, *but the ray that starts from* $\boldsymbol{x} \in R_i^-$ *and passes through* $\boldsymbol{e}_i^-$ *will overlap with* R^+. *More formally,* $R_i^- = \{\boldsymbol{x} | \overline{\boldsymbol{x}\boldsymbol{e}_i^-} \cap R^+ = \emptyset \wedge \overrightarrow{\boldsymbol{x}\boldsymbol{e}_i^-} \cap R^+ \neq \emptyset\}$. *Given* n^- *negative examples, the negative region* R^- *is the union of the negative region for each negative example, i.e.,* $R^- = \cup_{i=1}^{n^-} R_i^-$.

From the definition, we know that R_i^- is a convex cone generated by the conical combination of the vectors from the positive examples to the given negative example, i.e., $\overrightarrow{e_j^+ e_i^-}$ $(j = 1, \ldots, n^+)$. The red triangle in Fig. 3(a) depicts such a convex cone. However, the union of $R_i^-, i = 1, 2, \ldots$ is non-convex. For example, the union of the five red polygons in Fig. 3(b) is non-convex. Given more labeled examples, the result of the union will be more accurate for approximating the true negative region, which is outside the ellipse. We prove the following property of the negative region:

Proposition 3.2 *All points in the negative region* R^- *are negative.*

Definition 3.3 (Uncertain Region) *Denote the data space as* $\mathbb{R}^d$, *the uncertain region* $R^u = \mathbb{R}^d - R^+ - R^-$.

Formally, the polytope model makes a decision about an example $\boldsymbol{x}$ based on the following decision function, which takes values in $\{-1, 0, 1\}$ corresponding to R^-, R^u, and R^+ defined above:

$$F_{\mathcal{D}}(\boldsymbol{x}) = 1 \cdot \mathbb{1}(\boldsymbol{x} \in R^+) - 1 \cdot \mathbb{1}(\boldsymbol{x} \in R^-). \quad (1)$$

Our work also supports the case that the negative region of the query is convex ($\mathbf{Q}_c^-$). We can simply switch the above definitions such that we build a convex polytope for the negative region, and a union of convex cones for the positive region, one for each positive example. We also offer a test at the beginning of the data exploration process to choose between two polytope models, $Q \in \mathbf{Q}_c^+$ or $Q \in \mathbf{Q}_c^-$. The details are deferred to Section 4 where we offer a test procedure for all the assumptions made in the work.

Three-Set Metric. Our goal is not only to provide a new data space model, as described above, but also to design a new learning algorithm that enables a provable bound on the model accuracy. As stated before, our accuracy measure is F1-score. Formally, F1-score is evaluated on a *test set* $D_{test} = \{(\boldsymbol{x}_i, y_i)\}$, where $\boldsymbol{x}_i$ denotes a database object and y_i denotes its label according to the classification model. Then F1-score is defined as:

$$\text{F1-score} = 2 \cdot \frac{precision \cdot recall}{precision + recall},$$

where *precision* is the fraction of points returned from D_{test} by the model that are positive, and *recall* is the fraction of positive points in D_{test} that are returned by the model.

However, capturing F1-score in our data exploration procedure is difficult because we do not have such a labeled test set, D_{test}, available. We cannot afford to ask the user to label more to produce one since the user labor is an important concern. To bound the F1-score with limited labeled data, our idea is to run our polytope model, $F_{\mathcal{D}} : \mathbb{R}^d \to \{-1, 0, 1\}$, on an evaluation set. We define the *evaluation set* D_{eval} as the projection of D_{test} without labels y_i's. Then for each data point in D_{eval}, depending on which region it falls into, D_{eval} can be partitioned into three sets accordingly, denoted as D^+, D^- and D^u. We can compute a metric from the number of data points in the three sets, as follows.

Definition 3.4 (Three-Set Metric) *Denote* $D^+ = D_{eval} \cap R^+$, $D^- = D_{eval} \cap R^-$, $D^u = D_{eval} \cap R^u$, *and* $|S|$ *means the size of set* S. *At a specific iteration of exploration, the three-set metric is defined to be* $\frac{|D^+|}{|D^+|+|D^u|}$.

We will prove shortly that this metric is a lower bound of F1-score evaluated on D_{test}. As more labeled examples are provided, data points will be moved from D^u to either D^+ or D^-. Eventually, with enough training data the uncertain set shrinks to an empty set and the Three-Set Metric rises to 100%, reaching convergence.

3.2 Dual-Space Model and Algorithm

We now propose a new algorithm for interactive data exploration by exploiting models from two spaces, including our data space model $F_{\mathcal{D}}$ with the Three-Set Metric, and a classification model $F_{\mathcal{V}}$ with uncertainty sampling derived from the version space.

We first define the dual-space model (DSM) by considering two functionalities of a model for data exploration.

(1) *Prediction*: Given an unlabeled example, DSM predicts the class label first based on the data space model $F_{\mathcal{D}}$. If the example falls in the positive region R^+, it is predicted to be positive; if it falls in the negative region R^-, it is predicted to be negative. If the example falls in the unknown region R^u, then the classification model $F_{\mathcal{V}}$ is used to predict the example to be positive or negative.

(2) *Sampling*: In the active learning framework, the model is also used to guide the choice of the next example for labeling such that the new label can lead to significant improvement of the model accuracy. As discussed in Section 2, active learning uses an uncertainty sampling method, $\mathcal{S}_{\mathcal{V}}$, for a given classification model to choose the next example. However, directly application of uncertainty sampling to our dual-space model raises a problem: the example chosen by $\mathcal{S}_{\mathcal{V}}$ may fall in the known positive or negative region of our data space model $F_{\mathcal{D}}$, hence wasting computing resources on such examples. In our work, we propose an uncertainty sampling method that is restricted to the unknown region of our data space model, denoted as $\mathcal{S}_{\mathcal{D}}$. However, if we sample only from the unknown region of our data space model, we may not get a representative sample to train the classifier. Therefore, we use a sampling ratio, γ, to alternate between the sampling methods, $\mathcal{S}_{\mathcal{D}}$ and $\mathcal{S}_{\mathcal{V}}$. For instance, when $\gamma = 1/3$, in each iteration we use $\mathcal{S}_{\mathcal{D}}$ with probability 1/3 and use $\mathcal{S}_{\mathcal{V}}$ otherwise.

Now we present the full algorithm for interactive data exploration, as shown in Algorithm 1. The input is the database D, a positive example $\boldsymbol{x}^+$, a negative example $\boldsymbol{x}^-$, a user-defined accuracy threshold λ, and the sampling ratio γ. First, we extract an evaluation dataset D_{eval} from the database D (line 1). For now, let us assume D_{eval} to be D. Then we initialize data structures, setting the unknown partition of the evaluation dataset to be D_{eval}. The algorithm next goes through iterative exploration.

Lines 8-14 update our DSM model. The data space model, R^+ and R^-, is updated with the newly labeled example(s) by the user (lines 8-9). This step incrementally updates our convex polytope for R^+ and the union of convex polytopes for R^- based on computational geometry [3]. Afterwards, the corresponding partitions of D_{eval} are incrementally updated (line 10). In this step, some examples are removed from the unknown partition D^u and placed to the positive partition D^+ or the negative partition D^-. The accuracy is then estimated using the Three-Set Metric. We also keep track of the labeled and unlabeled examples using $D_{labeled}$ and $D_{unlabeled}$. We use the labeled examples to train a classifier, that is, the version space model in our DSM.

Then lines 15-28 implement uncertainty sampling using DSM. With probability γ, we perform uncertainty sampling from a pool that is restricted to the unknown partition of the evaluation set, D^u. Then the example chosen by uncertainty sampling is labeled by the user. With probability $1 - \gamma$, we perform uncertainty sampling from a pool that is a subsample of all unlabeled examples in the database. Then the example chosen by uncertainty sampling is first run through our data space model, (R^+, R^-), to see if it falls in the positive or negative region and hence can be labeled directly by the model. Otherwise, it will be labeled by the user.

Then the algorithm proceeds to the next iteration. It repeats until it has met the user accuracy requirement based on the lower bound offered by our Three-Set Metric, or reached the maximum of iterations allowed (line 29). Finally, we run the DSM model over the database to retrieve all tuples predicated to be positive. For the `subsample` procedures used in the algorithm, we defer their details to Section 3.4.

3.3 Lower Bounds of F1-score

Given how the DSM algorithm works, we now present formal results on the model accuracy achieved by DSM.

Exact Lower Bound. We begin with an exact lower bound of our accuracy measure, the F1-score.

Theorem 3.1 *The Three-Set Metric evaluated on* D_{eval} *is a lower bound of the F1-score if* DSM *is evaluated on* D_{test}.

The lower bound of DSM has several features. First, it is an *exact* lower bound throughout the exploration process for any evaluation set D_{eval}. Second, the metric is *monotonic*

Algorithm 1 Dual-Space Algorithm for Convex Queries

Input: database D, a positive example $\boldsymbol{x}^+$, a negative example $\boldsymbol{x}^-$, accuracy threshold λ, sampling ratio γ

1: $D_{eval} \leftarrow$ `subsample`(D, n)
2: $R^+ \leftarrow \emptyset$, $R^- \leftarrow \emptyset$
3: $D^+ \leftarrow \emptyset$, $D^- \leftarrow \emptyset$, $D^u \leftarrow D_{eval}$
4: $D_{labeled} \leftarrow \{\ \boldsymbol{x}^+, \boldsymbol{x}^-\}$
5: $D_{unlabeled} \leftarrow D \setminus D_{labeled}$
6: $D_{labeled\_by\_user} \leftarrow D_{labeled}$, $D_{labeled\_by\_dsm} \leftarrow \emptyset$
7: **repeat**
   // building the Dual-Space model:
8: **for** $\boldsymbol{x} \in D_{labeled\_by\_user}$ **do**
9: $(R^+, R^-) \leftarrow$ `updateRegion`$(R^+, R^-, \boldsymbol{x})$
10: $(D^+, D^-, D^u) \leftarrow$ `threeSets`$(R^+, R^-, D^+, D^-, D^u)$
11: $accu \leftarrow$ `threeSetMetric`(D^+, D^-, D^u)
12: $D_{labeled}$.`append`$(D_{labeled\_by\_user} \cup D_{labeled\_by\_dsm})$
13: $D_{unlabeled}$.`remove`$(D_{labeled\_by\_user} \cup D_{labeled\_by\_dsm})$
14: $classifier \leftarrow$ `trainClassifier`$(D_{labeled})$
   // uncertainty sampling:
15: $D_{labeled\_by\_user} \leftarrow \emptyset$, $D_{labeled\_by\_dsm} \leftarrow \emptyset$
16: **if** $rand() \leq \gamma$ **then**
17: $pool \leftarrow$ `subsample`(D^u, m)
18: $\boldsymbol{x} \leftarrow$ `getNextToLabel`$(pool, classifier)$
19: $D_{labeled\_by\_user} \leftarrow$ `getUserLabel`$(\boldsymbol{x})$
20: **else**
21: $pool \leftarrow$ `subsample`$(D_{unlabeled}, m)$
22: $\boldsymbol{x} \leftarrow$ `getNextToLabel`$(pool, classifier)$
23: **if** $\boldsymbol{x} \in R^+$ **then**
24: $D_{labeled\_by\_dsm} \leftarrow (\boldsymbol{x}, 1)$
25: **else if** $\boldsymbol{x} \in R^-$ **then**
26: $D_{labeled\_by\_dsm} \leftarrow (\boldsymbol{x}, -1)$
27: **else**
28: $D_{labeled\_by\_user} \leftarrow$ `getUserLabel`$(\boldsymbol{x})$
29: **until** $accu \geq \lambda$ **or** `reachedMaxNum`()
30: `finalRetrieval`$(D, (R^+, R^-), classifier)$

in the sense that points in the uncertain region D^u can be moved to the positive or negative region later, but not vice versa, and the metric goes to 1 when $D^u = \emptyset$. If the metric is above the desired accuracy threshold at some iteration, it is guaranteed to be greater than the threshold in later iterations, so we can safely stop the exploration.

Approximate Lower Bound. When D_{eval} is too large, we employ a sampling method to reduce the time to evaluate the Three-Set Metric. Let p and q be the true proportions of the positive and negative examples in D_{eval}, i.e., $p = |D^+|/|D_{eval}|$ and $q = |D^-|/|D_{eval}|$. Then the Three-Set Metric is $b = \dfrac{p}{1-q}$. Let $\hat{p}$ and $\hat{q}$ be the observed proportions of the positive and negative examples in a random draw of n examples from D_{eval}, and let $X_n = \dfrac{\hat{p}}{1-\hat{q}}$. Our goal is to find the smallest sample size n such that the error of the estimation X_n from the exact Three-Set Metric is less than δ with probability no less than λ. That is, $\Pr(|X_n - b| < \delta) \geq \lambda$.

The following theorem helps us find the lower bound of n.

Theorem 3.2 *$sup_\epsilon \| Pr(\sqrt{n}|X_n - b| < \epsilon) - \left(2\Phi(\dfrac{\epsilon(1-q)}{\sqrt{p(1-p-q)}}) - 1 \right) \| = O(1/\sqrt{n})$ for any ϵ, where Φ is the cumulative distribution function of the standard Normal distribution.*

With the theorem, we can approximate the sample size by

$$2\Phi(\frac{\sqrt{n}\delta(1-q)}{\sqrt{p(1-p-q)}}) - 1 \geq \lambda.$$

Since $p(1-p-q)/(1-q)^2 \leq 1/4$, it is sufficient for n to satisfy $2\Phi(2\sqrt{n}\delta) - 1 \geq \lambda$ and therefore $n \geq \left(\Phi^{-1}\left(\frac{\lambda+1}{2}\right)\right)^2/(4\delta^2)$.

3.4 Optimization of Sampling Methods

Sampling for creating the evaluation set (OPT1). In line 1 of Algorithm 1, we construct an evaluation set D_{eval} to develop a lower bound of the DSM model. Ideally, we want D_{eval} to be as large as the entire database D. For efficiency, we can only afford to have a memory-resident sample. The analysis in Theorem 3.2 indicates that we can choose a sample of size n from the database, and achieve an approximate lower bound of the F1-score of our DSM algorithm when evaluated on the database. The sample, denoted as $\tilde{D}_{eval}$, will expedite line 10 of Algorithm 1. For example, the SDSS database used in our experiments contains 190 million tuples. Denote the true lower bound as b, when $n = 50k$, our approximate lower bound $\hat{b}$ approximates b by $\epsilon \leq .005$ with probability 0.975 or higher. When $n = 1.9$ million, $\hat{b}$ approximates b by $\epsilon \leq .001$ with probability 0.994 or higher.

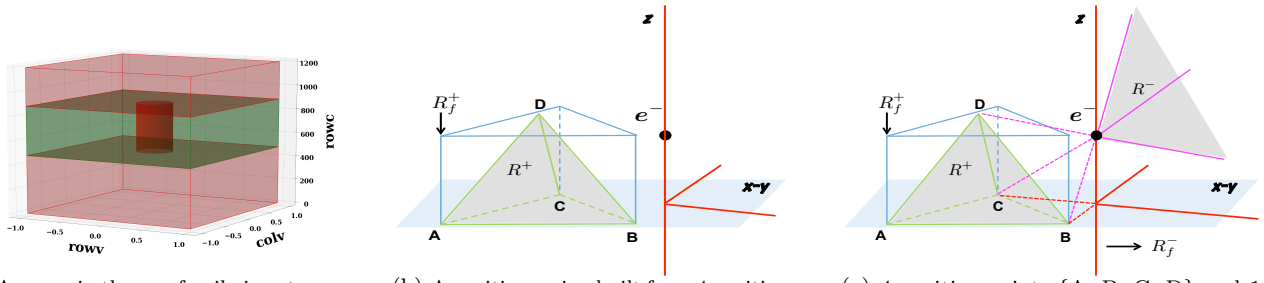
Sampling for pool-based active learning (OPT2). Algorithm 1 contains two `subsample` procedures for pool-based uncertainty sampling, that is, choosing the most uncertain example from the pool for labeling next. The `subsample` routine in line 17 creates a pool of unlabeled examples from the uncertain partition of the evaluation set, D^u, which is memory-resident. This can be implemented using reservoir sampling. The `subsample` routine in line 21, however, creates a pool, from the unlabeled portion of the entire database, which is large and not materialized. Our work offers a sampling-based optimization to avoid scanning the unlabeled database in each iteration. Due to space constraints the interested reader is referred to [22] for details.

4. FACTORIZATION

Although DSM can reduce user labeling effort and offer better accuracy than traditional active learning, increased dimensionality will make the volume of its uncertain region grow fast and degrade its performance. To handle this issue, we leverage the property of *conjunctive queries* to factorize a high-dimensional data space into a set of low-dimensional spaces. By running the polytope model in each low-dimensional space, we can "simulate" DSM in the high-dimensional space with improved performance. This extension, denoted as DSM$_F$, may require user labels of examples in some subspaces: If the user label is positive for an example, the label in each subspace is inferred to be positive. However, if the user label is negative for an example, she will be asked to specify the subset of attributes (and the subspaces thereof) that lead to the negative label. Since the user has gone through thinking when deciding the label, specifying a subset of attributes that lead to the negative label can be facilitated via a graphical interface such as in [11].

Factorization for DSC queries. Formally, factorization concerns a user interest query Q defined on an attribute set $\boldsymbol{A}$ of size d, and can be written in the conjunctive form, $Q_1 \wedge \ldots \wedge Q_m$. Each Q_i, $i \in [1 \ldots m]$, uses a subset of attributes $\boldsymbol{A}_i = \{A_{i1}, \cdots, A_{id_i}\}$. The family of attribute sets, $\boldsymbol{A} = (\boldsymbol{A}_1, \cdots, \boldsymbol{A}_m)$, is pairwise disjoint with $d = \sum_{i=1}^m d_i$, and is called the *factorization structure*.

In practice, we can derive the factorization structure from available data in a database. It is a partitioning of the database attributes with two properties: 1) Each attribute



(a) A query in the DSC family is not convex in either the positive region (green) or the negative region (red).

(b) A positive region built from 4 positive points, {A, B, C, D}

(c) 4 positive points {A, B, C, D} and 1 negative point (e^-), where e^- is negative in the (x-y) subspace and positive in the (z) subspace

Figure 4: Illustration of factorization when 3D space (x-y-z) is factorized into two subspaces (x-y) and (z).

can appear in only one partition. 2) If several attributes may be used in the same predicate (e.g., the positional attributes *rowc-colc*, or the velocity attributes *rowv-colv* from SDSS), they must be assigned to the same partition. If a query trace is available to show how attributes are used in predicates, e.g., individually or in a pair, we can run a simple algorithm over the query trace: Initially assign each attribute to its own partition. As the query trace is scanned, two partitions are merged if a predicate uses attributes from the two partitions. At the end, all the remaining partitions become the factorization structure $\boldsymbol{A}$. Even if a query trace is not available, it is not unreasonable to assume that by working with domain experts, it is possible to extract a reasonable factorization structure that reflects how attributes are used based on their semantics, e.g., the velocity attributes *rowv-colv* are often used in the same predicate.

DSC Queries. We next define a class of user interest queries that DSM$_F$ supports with benefits over active learning: that is, $\forall i = 1, \ldots, m$, either the positive region in the i^{th} subspace, defined as $\{\boldsymbol{x}_i \in \mathbb{R}^{|\boldsymbol{A}_i|} | Q_i(\boldsymbol{x}_i) > 0\}$, is convex (in the $\mathbf{Q}_c^+$ class), or the negative region, $\{\boldsymbol{x}_i \in \mathbb{R}^{|\boldsymbol{A}_i|} | Q_i(\boldsymbol{x}_i) < 0\}$ is convex (in $\mathbf{Q}_c^-$). We call such queries *Decomposable Subspatial Convex* (DSC) queries.

DSC allows us to combine a subspace whose positive region is convex with another subspace whose negative region is convex. However, the global query is **not** necessarily convex in either the positive or negative region. Fig. 4(a) shows an example query, $Q = x^2 + y^2 > 0.2^2 \wedge 480 < z < 885$, which combines the ellipse pattern in Fig. 2(d), whose negative region is convex, with the line pattern in Fig. 2(a), whose positive region is convex. In the 3D space, the negative region (marked in red) is the union of the red cylinder in the middle of the figure and the red rectangles above and below the cylinder, while the positive region (marked in green) is the complement of it. Neither of the positive nor the negative region of Q is convex although it belongs to DSC.

p-DSC Queries. We also support a superset of DSC queries, where the convex assumption holds only in some subspaces. We call this generalization *partial factorization* or p-DSC.

As a special case, if none of the subspaces permits the convexity property, we call such queries *Zero* DSC or 0-DSC.

Polytope model with factorization and DSM$_F$. Given the factorization structure, the polytope model runs in each subspace where the subspatial convexity is deemed true. First, in the i^{th} subspace defined by $\boldsymbol{A}_i$, Q_i's positive and negative regions, denoted as R_i^+ and R_i^-, are built according to Definition 3.1 and 3.2, but based on the "positive" and "negative" labels of projected examples for Q_i, as described above. Then we build the positive and negative regions of Q from the positive and negative regions in the subspaces via the *conjunctive property*:

$$R_f^+ = \times_{i=1}^m R_i^+, \quad R_f^- = \left(\times_{i=1}^m (R_i^-)^c\right)^c \tag{2}$$

where $\times$ denotes the Cartesian product between two sets, and R^c denotes the complement of set R. Then the uncertain region of Q is, $R_f^u = \mathbb{R}^d - R_f^+ - R_f^-$.

Next we formally define the decision function for the polytope model with factorization. In each subspace defined on $\boldsymbol{A}_i$, let $F_{\boldsymbol{A}_i} : \mathbb{R}^{|\boldsymbol{A}_i|} \to \{-1, 0, 1\}$ be the decision function that divides the subspace into three disjoint regions corresponding to the negative, unknown, and positive regions, respectively. As in (Eq. 1),

$$F_{\boldsymbol{A}_i}(x) = 1 \cdot \mathbb{1}(\boldsymbol{x} \in R_i^+) - 1 \cdot \mathbb{1}(\boldsymbol{x} \in R_i^-). \tag{3}$$

For p-DSC queries, if the subspatial convexity is deemed not true in a subspace, the value of its decision function is a constant zero.

The global decision function for the polytope model with factorization over the entire data space is then

$$F_{\mathcal{D}_f}(\boldsymbol{x}) = \min_{i=1}^m F_{\boldsymbol{A}_i}(\boldsymbol{x}_i) \tag{4}$$

where $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d)$, and $\boldsymbol{x}_i$ denotes the projection of $\boldsymbol{x}$ on the i^{th} subspace defined by $\boldsymbol{A}_i$.

Finally, consider the dual-space model. Let DSM$_F$ be DSM in §3.2 with the polytope data space model $F_{\mathcal{D}}$ replaced by the polytope model with factorization $F_{\mathcal{D}_f}$. Then the dual-space decision function of DSM$_F$, $H : \mathbb{R}^d \to \{-1, 1\}$, is:

$$H(\boldsymbol{x}) = \begin{cases} F_{\mathcal{D}_f}(\boldsymbol{x}), & F_{\mathcal{D}_f}(\boldsymbol{x}) \neq 0 \\ F_{\mathcal{V}}(\boldsymbol{x}), & \text{otherwise} \end{cases} \tag{5}$$

where $F_{\mathcal{V}}$ is the classification model. H is our final prediction model returned to approximate the user interest query. For 0-DSC queries, DSM$_F$ simply runs traditional active learning.

Illustration. Before presenting the formal results, we illustrate the intuition that factorization allows us to construct the positive and negative regions (R_f^+, R_f^-) as supersets of (R^+, R^-), hence reducing the unknown region and offering better accuracy. Fig. 4(b) shows four positive points A, B, C, D when the 3D space (x-y-z) is factorized into two subspaces (x-y) and (z). It depicts the positive region R^+ as the pyramid shaded in grey and marked by the green lines. When we factorize R^+ into (x-y) and (z) planes, we have R_{xy}^+ as a triangle marked ABC and R_z^+ as a line segment

projected onto z. Then we construct R_f^+ from the triangle and the line segment based on Eq. 2, we obtain a prism marked by the blue lines, which is much bigger than R^+.

Fig. 4(c) shows a negative point e^- and the negative region constructed for it. R^- is a convex cone shaded in grey and bounded by the solid purple rays emitting from e^-. When e^- is projected onto (x-y), it is negative and defines a convex cone R_{xy}^- marked by the two solid red lines in the (x-y) plane. When e^- is projected onto z, it lies in the positive region. According to Eq. 2, the new negative region R_f^- extends the convex cone R_{xy}^- by all possible values of z, resulting in a geometric shape enclosed by the three solid red lines in the figure. Again, the new R_f^- is much larger than R^-.

Formal results. For both DSC and p-DSC queries, we offer formal results of the polytope model with factorization.

Proposition 4.1 *All points in the positive region R_f^+ are positive and $R^+ \subseteq R_f^+$ at each iteration of the data exploration process.*

Proposition 4.2 *All points in the negative region R_f^- are negative and $R^- \subseteq R_f^-$ at each iteration of the data exploration process.*

Proposition 4.3 DSM$_F$ *improves the Three-Set Metric, the lower bound of the model accuracy in F1-score, over* DSM.

Proposition 4.1 and Proposition 4.2 state that the positive and negative regions constructed via factorization, (R_f^+, R_f^-), are a superset of (R^+, R^-) that the original DSM offers. Hence, the lower bound of F1-score built from (R_f^+, R_f^-) is higher than that from (R^+, R^-) based on Def. 3.4.

Testing assumptions of dsm$_F$. Factorization replies on two assumptions, which our system will test when a user is performing data exploration online. The first assumption is that we have a correct factorization structure, $\boldsymbol{A} = (\boldsymbol{A}_1, \cdots, \boldsymbol{A}_m)$, for a database derived from its query trace or schema. The second assumption is that user interests take the form a conjunctive query (CQ). In fact, our system offers a simple extension of DSM to handle categorical attributes, which is left to [22] in the interest of space. With this extension, the class of user interest queries is an *extended class of conjunctive queries*, in the form of $P_1 \wedge P_2 \wedge \ldots$, such that either P_i is an individual predicate, or P_i is defined on a categorical attribute A and can take the disjunctive form, $(A = 1) \vee (A = 2) \vee ...$

Test Procedure. When either the factorization structure or the CQ assumption is wrong, we start to see conflicting examples in a polytope model for a specific subspace, i.e., a positive example labeled by the user appears in the negative region or vice versa. Based on this, our system uses the following procedure to test online both assumptions behind DSM$_F$. At the beginning of data exploration, we build two polytope models for each subspace, one under the assumption that the positive region is convex, $\mathbf{Q}_c^+$, the other under the assumption that the negative region is convex, $\mathbf{Q}_c^-$. Over iterations, we count the number of conflicting examples of these two polytope models, and use a threshold, T, to turn off a polytope model if its count exceeds T. If both polytope models remain active in a given iteration, the one with the smaller count will be used; in the case of a tie, the model for $\mathbf{Q}_c^+$ will be used as more query patterns belong to this class. When both polytope models for a subspace are turned off, we use partial factorization until they are turned off for all subspaces, when we resort to the classifier. The test procedure also allows DSM$_F$ to handle noisy user labeling by adjusting the threshold T, the number of conflicting examples that can be tolerated by each polytope model.

5. ONLINE FEATURE SELECTION

The user exploration task may start with more attributes than those included in the final learned model (user interest query). The attributes that are not included in the final model are noise in data exploration and cause slow convergence of the model. To remove such noisy attributes, we employ both offline dimensionality reduction on the database and online feature selection techniques.

We examined dimension reduction methods including PCA for offline compression, and Random forests (RF) and Gradient boosting regression trees (GBRT) for online feature selection. Since these are standard methods, we leave their description and evaluation to [22]. We found that GBRT works best for reducing dimensions. Therefore, we outline how GBRT is used for online feature (attribute) selection in our work. In each iteration of data exploration, we feed all labeled examples to the GBRT learner, and ask the learner to return the top-k features that are deemed most important in learning. Then we build the classifier using these features and select the next example for labeling. We repeat the two steps in each iteration until the choice of top-k features stabilizes. Then we build the full DSM$_F$ model until it converges. However, we observe two limitations of this approach:

Unbalanced training data. GBRT is very sensitive to unbalanced classes, that is, when the training data is dominated by the negative examples. This is common in data exploration because the true user interest is often a highly selective query. We draw upon two insights in this work to mitigate the imbalance problem. First, when applicable, our DSM$_F$ algorithm already maintains a list of positive examples from the evaluation set and we can add them to make the training data balanced. Second, before DSM$_F$ accumulates enough positive examples, we can also boost GBRT using synthetic positive data: if the user interest has a convex shape, as long as there are two positive examples, we can draw points from the line that connects these two examples, and treat these points as synthetic positive examples to balance the training data.

How many features to select? Another issue is that we do not know how many features to select, or the exact value of top-k. The user does not offer this information a priori. Choosing the right value of k is nontrivial because GBRT may not have high confidence for selecting the correct features in earlier iterations. To formalize the notion of confidence, we utilize the *feature importance scores* provided by GBRT. GBRT is a linear combination of decision tree base-learners, where each decision tree intrinsically selects features for splitting nodes. For each decision tree, the importance score of a feature is computed as weighted sum of impurity decreases for all the nodes where the feature is used [6]. Then this score is averaged over all trees. Given feature importance scores, we propose two strategies to characterize "sufficient confidence" of GBRT for feature selection.

Proportion of nonroot trees: The intuition is that all decision trees must be weak learners and hence find some features useful in distinguishing the positive class from the negative

Table 1: Query templates with different selectivity values

Query template
Q1 (rectangle): $rowc \in [a_1, a_2] \wedge colc \in [b_1, b_2]$
Q2 (ellipse): $((rowc - a_1)/b_1)^2 + ((colc - a_2)/b_2)^2 < c^2$
Q3 (rectangle): $ra \in [a_1, a_2] \wedge dec \in [b_1, b_2]$
Q4 (outside a circle): $rowv^2 + colv^2 > c^2$
Q5: 4D queries combining two queries from Q1-Q4
Q6: 6D queries combining three from Q1-Q4
Q7: 6D-11D queries with 4-9 irrelevant attributes
Q8: 4D, $(x_1 > a + b \cdot x_2) \wedge (x_3 + c \cdot \log_{10} x_4^2 < d)$

class. Based on this intuition, we wait until the iteration where all trees become nonroot trees, hence likely to be weak learners. Then we believe that the confidence of GBRT has reached a sufficient level.

Entropy of Importance Scores (EIS): The next intuition is that we prefer to have a lower entropy of the importance scores across all the features. That is, the distribution of importance scores departs from a uniform distribution and becomes concentrated. Based on this, our second strategy is to wait until EIS has dropped significantly below the expected entropy of uniform importance scores, i.e., the importance scores of some features really stand out. Then we think that the confidence of GBRT has been sufficient.

Adaptive Feature Selection: We next devise adaptive strategies to decide top-k features, depending on whether the current iteration has reached the point of sufficient confidence for feature selection, based on any of the above strategies. Before reaching this point, we perform conservative feature filtering: we select top-k features that account for 50% of the total feature importance scores. After GBRT reaches the point of sufficient confidence, we perform aggressive feature selection by scanning the ranked list of feature importance scores and choosing the top-k features such that the k^{th} feature and the $k + 1^{th}$ feature have the largest gap in the ranked list. When GBRT chooses the same top-k features for 10 iterations, we consider the choice of relevant features stable and use them to build DSM$_F$.

6. EXPERIMENTAL EVALUATION

We implemented all of our proposed techniques in a Java-based prototype for data exploration, which connects to a PostgreSQL database. In this section, we evaluate our techniques and compare to recent active learning algorithms [5, 15], active search [16], and explore-by-example systems, Aide [12, 13] and LifeJoin [9].

Datasets: Our evaluation used two datasets. (1) SDSS (190 million tuples) contains the "PhotoObjAll" table with 510 attributes. By default, we used 1% sample (1.9 million tuples, 4.9GB) to create an evaluation set for DSM and for pool-based uncertainty sampling – our formal results in Section 3.4 allowed us to use the 1% sample for data exploration, yet with bounded difference of $\epsilon \leq .001$ from the accuracy achieved over the full database with probability $\geq$0.994. (2) Car database (5622 tuples): this small dataset is used for our user study because it is more intuitive for users to perform explorative analytics. We defer a detailed discussion to §6.3.

User Interest Queries: We extracted 8 query templates from the SDSS query release [38] to represent user interests. They allow us to run *simulations* of user exploration sessions, as in [12, 13], by using the true query answers as the proxy for user labeling. The 8 templates are shown in Table 1. Each template is instantiated with different constants to vary query selectivity in [0.01%, 10%]. In particular, Q1-Q3 represent patterns that are convex in the positive region ($\mathbf{Q}_c^+$). Q4 retrieves tuples outside a circle, hence in the $\mathbf{Q}_c^-$ class. Q5-Q7 combine these attributes, optionally with irrelevant attributes, for scalability tests. Q8 includes a predicate on $x1$ and x_2 that belongs to $\mathbf{Q}_c^+$, and a log predicate on x_3 and x_4 that belongs to $\mathbf{Q}_c^-$ if $x_4 > 0$ or is non-convex if $x_4 \in \mathbb{R}$.

6.1 Dual-Space Algorithm with Factorization

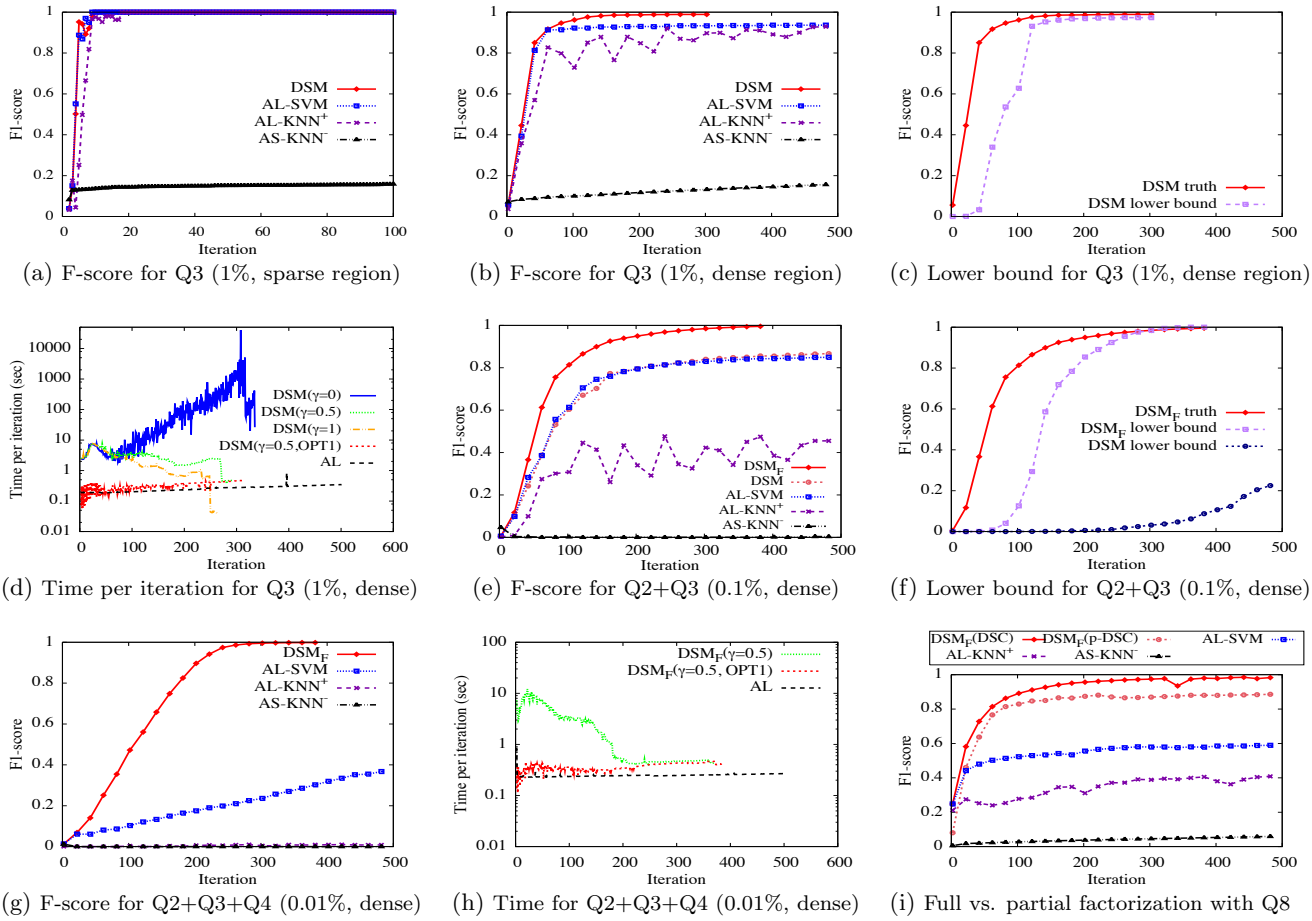
We evaluate our Dual-Space Model (DSM) and compare it to two ML techniques: (*i*) Active Learning (AL) runs uncertainty sampling [5,15] to obtain labeled examples for building a classifier, which is the version space part of DSM. We run AL with an SVM or a standard kNN classifier (kNN$^+$). (*ii*) Active Search (AS) [16] also follows an iterative procedure of asking the user to label an example and training a new model to guide the selection of the next example, but uses a strategy to maximize the number of positive examples in the set of selected examples, not classification accuracy. It uses a special variant of kNN classifier to enable optimization, denoted as kNN$^-$. All tests were run up to 500 labeled examples or when the lower bound of F1-score reaches 99%. In all plots, the x-axis is the number of labeled examples.

Expt 1 (2D queries): We run 2D queries from templates Q1 to Q4. Since the results show similar trends, we show the F1-score, lower bound, and time measurement for Q3 (1%) in Fig. 5(a)-5(d).

Regarding accuracy, DSM outperforms AL, and AL outperforms AS. (1) A factor that affects performance is the data distribution around the decision boundary B of the user interest query. If B falls into a sparse region, separating the positive and negative classes is relatively easy for DSM and AL. Fig. 5(a) shows that for Q3 whose decision boundary is in a sparse region, DSM and AL-SVM converge within 10 labeled examples. However, AS performs poorly, with F1-score less than 20%. The reason is that AS compromises recall by searching close to existing positive examples. In contrast, DSM and AL aims to maximize classification accuracy by sampling the most uncertain region of the model, e.g., close to the decision boundary, hence offering better F1-score. (2) If B falls into a dense data region, learning an approximate $\tilde{B}$ that can accurately divide all unlabeled data points requires more labeled examples. To reach 95% accuracy, DSM requires 100 labeled examples for Q3 and 90 examples on average for Q1-Q4. AL works better with the SVM than the kNN classifier, but even AL-SVM cannot reach 95% for Q3 or most other workloads. Finally, AS performs much worse than AL-KNN$^+$ while both use a kNN classifier.

DSM further offers a lower bound in F1-score. Fig. 5(c) shows that the lower bound is quite close to the true F1-score.

The time cost of DSM depends on the sampling method. Recall from Algorithm 1 that with probability γ, it samples from the unknown partition of the polytope model, D^u; otherwise, it does so from a subsample of the unlabeled examples in the database, $D_{unlabeled}$. γ=0 leads to high time costs because the examples retrieved from $D_{unlabeled}$ may repeatedly fall in the positive or negative region of the polytope model, wasting resources with no new information. γ=1 and γ=0.5 significantly reduce the time cost, with γ=0.5 offering slightly better accuracy due to balanced sampling. However, both settings exhibit a spike in time cost in the early phase, which is the time to build the polytope model the first time by scanning the entire evaluation set. Finally, we improve



(a) F-score for Q3 (1%, sparse region) (b) F-score for Q3 (1%, dense region) (c) Lower bound for Q3 (1%, dense region)

(d) Time per iteration for Q3 (1%, dense) (e) F-score for Q2+Q3 (0.1%, dense) (f) Lower bound for Q2+Q3 (0.1%, dense)

(g) F-score for Q2+Q3+Q4 (0.01%, dense) (h) Time for Q2+Q3+Q4 (0.01%, dense) (i) Full vs. partial factorization with Q8

Figure 5: DSM with factorization for 2D, 4D, and 6D queries, compared to Active Learning (AL) and Active Search (AS) algorithms

γ=0.5 with the optimization (OPT1) from Section 3.4, where we start with a smaller evaluation set to avoid the initial spike, but later switch to a larger evaluation set once its polytope model is built completely in the background. This optimization keeps the time cost per iteration within a second and will be used as the default algorithm.

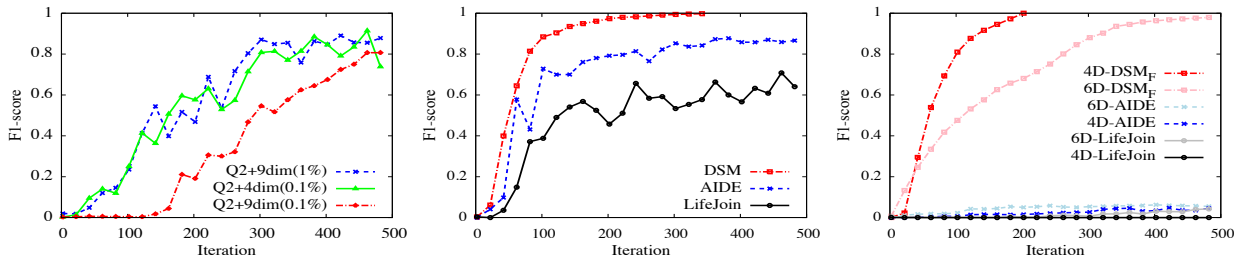
Expt 2 (4D-6D queries): We next show results of 4D-6D queries in dense regions as they present harder workloads. The main results of Fig. 5(e)-5(h) are: (1) Without factorization, DSM performs similarly to AL because the polytope model is dominated by the uncertain region. Then DSM$_F$ dramatically shrinks the uncertain region, and improves the lower bound and F1-score. (2) Consistently, DSM$_F$ outperforms AL, which further outperforms AS. Fig. 5(g) shows that for the 6D query, DSM$_F$ reaches 95% with 240 examples, AL-SVM cannot reach 40% with 500 examples, and AS has F-score close to 0. (3) DSM$_F$ can keep the time per iteration within 1-2 seconds, as shown in Fig. 5(h) for the 6D query.

Expt 3 (Partial factorization): We next generate two queries from Q8 with 7.8% selectivity (based on the constants used in SDSS). Q8.v_1 is in the DSC family because its positive region is convex in the ($x1$, x_2) subspace, and its negative region is convex in (x_3, x_4). Q8.v_2 is in the p-DSC family because it is non-convex in (x_3, x_4). Hence, DSM supports Q8.v_2 with partial factorization. Fig. 5(i) shows that DSM$_F$ works better for the DSC query than p-DSC query, as expected, but even partial factorization works much better than (i) AL, which does not reach 60% after 500 labeled examples, and (ii) AS, which cannot achieve more than 5% accuracy.

Expt 4 (Feature selection for 6D-11D queries): We now evaluate the optimizations proposed for GBRT as an online feature selection method in Section 5. To do so, we use the template Q7. Since the major results are similar, below we report results by extending Q2 with 4 irrelevant attributes (making it a 6D query) or with 9 irrelevant attributes (making the query 11D). Without feature selection, the F1-score is 0 for these queries. As Fig. 6(a) shows, feature selection improves the F1-score to above 80% for all of them. As stated earlier, GBRT is sensitive to selectivity and hence the combination of high dimensionality (11D) and low selectivity (0.1%) represents the hardest workload among the three.

6.2 Comparison to Alternative Systems

We next compare our system to two state-of-the-art explore-by-example systems: 1) LifeJoin [9] uses SVM for classification and active learning for seeking the next example for labeling. But it differs in the SVM implementation from our work. We implemented LifeJoin techniques as additional methods in our system. 2) Aide [12,13] uses decision trees as the classification model and customized methods for seeking the next example for labeling. We obtained the source code from the authors. When comparing these systems, we ex-



(a) Feature selection for Q7 (6D-11D) (b) DSM vs Aide and LifeJoin (2D) (c) DSM vs Aide and LifeJoin (4D, 6D)

Figure 6: Results for partial factorization, feature selection, and comparison to Aide and LifeJoin systems, and the user study.

clude the overhead to find the first positive example as these systems use different methods / assumptions to find them.

Expt 5: F1-score is reported in Fig. 6(b) for a 2D query, and in Fig. 6(c) for 4D and 6D queries. (1) For the 2D query, our system outperforms Aide, which further outperforms LifeJoin. Overall, LifeJoin is significantly worse in accuracy. (2) For the 4D query, Aide and LifeJoin drop to below 10% in accuracy, while our system achieves 99% within 200 iterations. For the 6D query, again Aide and LifeJoin fail to work. This observation remains for any query that we tried beyond 2D. This is due to the combination of low selectivity and high dimensionality in data exploration. Additional analysis of these systems is available in [22].

6.3 User Study using a Car Database

We conducted a user study by building a car database[3]. The database includes 5622 vehicles with 27 attributes such as the model, year, length, height, engine power, and retail price. Additional details on the database are given in [22]. Our study has two objectives: (1) build a query trace to understand the characteristics of data exploration tasks in this domain; (2) use this trace as the ground truth of user interests to evaluate our system.

To develop the query trace, we designed task scenarios with different price constraints and usage patterns, e.g., buying a car for everyday city commute, outdoor sports, an elderly in the family, or a small business. The 18 users in our study belong to two groups: the first 11 users are CS professors and graduate students, while the rest of 7 are non-technical people. We asked each user to find all the cars that meet the requirements of the assigned task so that he can provide a recommendation service to customers who belong to the given scenario. Each user proceeds in three phases: 1) Review the schema: Since this is a familiar domain, most users can understand the schema quickly. 2) Initial exploration: We next show sample data to help the user understand the data and materialize his preference. We also ask the user to come up with the first positive example via (a) naming a car that he already has in mind, or (b) reviewing a sample set pre-computed for the given task based on the price constraints, body type, year, etc., and finding one that appears relevant. Two users chose option (a) while the others chose option (b). 3) Iterative manual exploration: In the third phase, the user is asked to specify his interest precisely by (a) sending a SQL query to the database[4]; (b) reviewing a subset of the returned tuples; (c) going back to revise the query. The steps are repeated until the user is satisfied with all returned tuples of the final query.

First, we verify that the selectivities of all 18 queries are in the range of [0.2%, 0.9%]. They contain 117 predicates in total: 70 of them are defined on numerical attributes and are all convex in the positive region. The rest 47 use categorical attributes, for which the notion of convexity does not apply. All the queries are conjunctive; the only disjunction is applied to the categorical attributes.

Second, we evaluate our system by running simulations using these queries as the true user interest. While the queries involve 4 to 10 attributes, the classifier requires categorical attributes to be transformed using one-hot encoding, resulting in 4 to 418 features used by DSM. Table 2 shows in the "DSM" column family that DSM$_F$ achieve 99% for all queries, with a median of 10 labeled examples, despite the high-dimensionality. In contrast, the users manually wrote a series of queries, with a median of 12, and reviewed many tuples, with a median of 98. The results from two user groups are largely consistent with each other, with a small difference that non-technical people tend to specify simpler queries, hence easier for DSM to learn. Note that the initial cost of finding the first positive example, with a median of 10 tuples, can be added fairly to both exploration modes.

Third, the study verified our benefits over active learning (AL), which cannot reach 95% accuracy for most queries within 100 iterations and for 80% has a median of 29 labeled examples. Even if the decision boundary is often in a sparse region, high dimensionality makes AL lose performance.

Fourth, to test the robustness of DSM we run it with noisy user labels as described in Section 4. Any polytope model that has exceeded T=1 conflicting examples will be turned off, leading to partial factorization. To inject noisy labels, our intuition is that they are more likely to occur close to the boundary of the true pattern. 1) *Noise model*: we use a *Gaussian* noise model where a new example gets a wrong label with a probability that depends on its distance to the true boundary, i.e., the probability that its distance to the boundary of a perfect SVM classifier (with the margin size δ) is less than a sample from a Gaussian function with u=0 and $\sigma = \delta/2$. 2) *Non-uniform user behaviors*: we assume the first s (=5 or 10) examples to be noise free because they are drawn from the broad data space, hence likely to be far from from the true boundary, and the user tends to be more patient at the beginning. The last 9 columns of Table 2 show DSM and AL under noisy labels. DSM reaches 95% accuracy with a median of 11 labeled examples including 5 noise-free initial examples, while AL requires 36 labeled examples with 10 noise-free examples to achieve 80%. Out

[3]The content is extracted from http://www.teoalida.com/

[4]For the 7 non-techncial people, we offered help to translate their requirements to SQL but otherwise did not influence the users in data exploration.

Table 2: Results of the car database using Manual Exploration, DSM, Active Learning (AL). #A shows the number of attributes used in the user interest and #F shows the number of features (with onehot encoding) after transforming all categorical attributes for use by the classifier. For manual exploration, T_2 shows the number of tuples reviewed in initial exploration (the 2nd phase) for the user to find the first positive example; and T_3 shows those reviewed in iterative exploration (3rd phase). For DSM and AL, the algorithm marked by '-' never reached the desired accuracy within 100 iterations.

Q		#A	#F	Manual Exploration			AL			DSM			DSM 5 noise free			DSM 10 noise free			AL 10 noise free		
				T_2	T_3	#SQL	0.8	0.95	0.99	0.8	0.95	0.99	0.8	0.95	0.99	0.8	0.95	0.99	0.8	0.95	0.99
Q1	**Min**	4	12	0	35	8	8	9	9	4	4	5	4	4	5	4	4	5	8	9	9
∼	**Max**	8	418	18	412	49	-	-	-	14	23	26	-	-	-	14	-	-	-	-	-
Q11	**Mdn**	6	35	11	104	16	29	42	-	7	10	13	7	13	-	7	10	13	35	-	-
Q12	**Min**	4	4	0	50	4	9	15	15	3	5	6	3	5	6	3	5	6	9	16	-
∼	**Max**	10	51	24	260	15	-	-	-	7	11	12	-	-	-	7	11	12	-	-	-
Q18	**Mdn**	6	24	4	91	10	28	-	-	6	6	9	6	6	-	6	6	9	48	-	-
Global	**Mdn**	**6**	**30**	**10**	**98**	**12**	**29**	**-**	**-**	**6**	**9**	**10**	**6**	**11**	**-**	**6**	**9**	**10**	**36**	**-**	**-**

of the 105 subspatial polytope models built for 18 queries, 14% were turned off when s=5 and 6% were turned off when s=10. Partial factorization still outperforms AL for noisy labels due to fast convergence.

7. RELATED WORK

Data Exploration. *Faceted search* iteratively recommends query attributes for drilling down into the database, but the user is often asked to provide attribute values until the desired tuple(s) are returned [26, 34, 35] or offer an "interestingness" measure and its threshold [10]. Semantic windows [25] are pre-defined multidimensional predicates that a user can explore. These methods are different from our active learning approach. Recent work also supports time series data [32] or avoids false discoveries of statistical patterns [44] during interactive data exploration. Finding best database objects based on user preferences [40] assumes a numeric weight per database attribute, which is different from the active learning approach to discover the user interest on the fly.

Query by Example is a specific framework for data exploration. Earlier work on QBE focused on a visualization front-end that aims to minimize the user effort to learn the SQL syntax [23, 33]. Recent work [31] proposes exemplar queries which treat a *query* as a sample from the desired result set and retrieve other tuples based on similarity metrics, but for graph data only. The work [37] considers data warehouses with complex schemas and learns the minimal project-join queries from a few example tuples efficiently. It does not consider selection with complex predicates, which is a focus of our work. The work [24] helps users construct join queries for exploring relational databases, and [28] does so by asking the user to determine whether a given output table is the result of her intended query on a given database.

Query formulation has been surveyed in [8]. The closest to our work is LifeJoin [9], which we compared in Section 6.2. Query By Output (QBO) [42] takes the output of one query on a database, and constructs another query such that running these two queries on the database are instance-equivalent. Dataplay [2] provides a GUI for users to directly construct and manipulate query trees. It assumes that the user can specify the value assignments used in his intended query, and learns conjunctions of quantified Horn expressions (with if-then semantics) over nested relations [1].

Active Learning. Tong and Koller [41] provide a theoretical motivation on selecting new examples using the notion of a version space, but with unknown convergence speed. Related to our work is a lower bound on the probability of misclassification error on the unlabeled training set [7]. However, it relies on user labeling of an additional sample from the unlabeled pool, which is not required in our work. Recent papers [14, 18–20] offer probabilistic bounds for the classification error and sample complexity. Our work differs in that we focus on F1-score, which suits selective user interest queries (imbalanced classes in classification).

Most learning theory makes no assumptions on convex data/class distributions [21]. Clustering techniques can assume that data is clustered in convex sets [21], but address a different problem from ours. In Active Learning, convexity assumptions occur in the Version Space [4, 41], which is the set of classifiers consistent with training data. DSM can embrace any classifier developed through the version space, but also includes the new polytope model.

Active Search. Active search [16, 30, 43] aims to maximize the number of positive examples discovered, called the *Target Set Accuracy*, within a limited budget of user labeling effort. In comparison, our work aims to maximize the *F1-score* of the model learned to approximate the true user interest. We reported the performance difference from [16] in the previous section. The works [30, 43] use a kernel function to measure similarity of items in order to maximize the utility of a set of selected items. Such kernel methods have the smoothness requirement, i.e., similar items have similar utility values, and require training data to tune the kernel for each use case (user interest), which do not suit our problem setting.

8. CONCLUSION AND FUTURE WORK

We presented new algorithms for interactive data exploration in the active learning framework, which leverage the subspatial convex and conjunctive properties of database queries to overcome the slow convergence of active learning. Our results show that our DSM algorithm significantly outperforms active learning [5, 15], active search [16], and existing explore-by-example systems, Aide [12, 13] and LifeJoin [9], in accuracy and convergence speed, while maintaining the per-iteration time within 1-2 seconds. In future work, we will address inconsistent labeling by extending our DSM model to a probabilistic model, extend query patterns to multiple disjoint areas using exploration versus exploitation, and leverage data properties to further improve accuracy.

Acknowledgements. This work was funded in part by NSF under the grant IIS-1218524, Agence Nationale de la Recherche (ANR), and Université Paris-Saclay.

9. REFERENCES

[1] A. Abouzied, D. Angluin, C. H. Papadimitriou, J. M. Hellerstein, and A. Silberschatz. Learning and verifying quantified boolean queries by example. In *Symposium on Principles of Database Systems (PODS)*, pages 49–60, 2013.

[2] A. Abouzied, J. M. Hellerstein, and A. Silberschatz. Playful query specification with dataplay. *PVLDB*, 5(12):1938–1941, 2012.

[3] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483, Dec. 1996.

[4] K. Bellare, S. Iyengar, A. Parameswaran, and V. Rastogi. Active sampling for entity matching with guarantees. *ACM Transactions on Knowledge Discovery from Data*, 7(3):12:1–12:24, Sept. 2013.

[5] A. Bordes, S. Ertekin, J. Weston, and L. Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6:1579–1619, Dec. 2005.

[6] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[7] C. Campbell, N. Cristianini, and A. J. Smola. Query learning with large margin classifiers. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 111–118, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[8] A. Cheung and A. Solar-Lezama. Computer-assisted query formulation. *Foundations and Trends® in Programming Languages*, 3(1):1–94, June 2016.

[9] A. Cheung, A. Solar-Lezama, and S. Madden. Using program synthesis for social recommendations. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 1732–1736, New York, NY, USA, 2012. ACM.

[10] D. Dash, J. Rao, N. Megiddo, A. Ailamaki, and G. Lohman. Dynamic faceted search for discovery-driven analysis. In *CIKM*, pages 3–12, 2008.

[11] Y. Diao, K. Dimitriadou, Z. Li, W. Liu, O. Papaemmanouil, K. Peng, and L. Peng. AIDE: an automatic user navigation system for interactive data exploration. *PVLDB*, 8(12):1964–1967, 2015.

[12] K. Dimitriadou, O. Papaemmanouil, and Y. Diao. Explore-by-example: an automatic query steering framework for interactive data exploration. In *SIGMOD Conference*, pages 517–528, 2014.

[13] K. Dimitriadou, O. Papaemmanouil, and Y. Diao. Aide: an active learning-based approach for interactive data exploration. *IEEE Transactions on Knowledge and Data Engineering*, 28(11):2842–2856, 2016.

[14] R. El-Yaniv and Y. Wiener. Active learning via perfect selective classification. *Journal of Machine Learning Research*, 13(1):255–279, Feb. 2012.

[15] S. Ertekin, J. Huang, L. Bottou, and L. Giles. Learning on the border: Active learning in imbalanced data classification. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 127–136, New York, NY, USA, 2007. ACM.

[16] R. Garnett, Y. Krishnamurthy, X. Xiong, J. G. Schneider, and R. P. Mann. Bayesian optimal active search and surveying. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*, pages 843–850, 2012.

[17] B. Grünbaum. Convex polytopes. In *Convex Polytopes*. Springer-Verlag New York, 2 edition, 2003.

[18] S. Hanneke. Rates of convergence in active learning. *The Annals of Statistics*, 39(1):333–361, 02 2011.

[19] S. Hanneke. Theory of disagreement-based active learning. *Foundations and Trends® in Machine Learning*, 7(2-3):131–309, June 2014.

[20] S. Hanneke. Refined error bounds for several learning algorithms. *The Journal of Machine Learning Research*, 17(1):4667–4721, Jan. 2016.

[21] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[22] E. Huang, L. Peng, L. D. Palma, A. Abdelkafi, A. Liu, and Y. Diao. Optimization for active learning-based interactive database exploration. Technical report, 2018. `https://hal.inria.fr/hal-01870560`.

[23] B. E. Jacobs and C. A. Walczak. A Generalized Query-by-Example Data Manipulation Language Based on Database Logic. *IEEE Transactions on Software Engineering*, 9(1):40–57, 1983.

[24] M. Kahng, S. B. Navathe, J. T. Stasko, and D. H. P. Chau. Interactive browsing and navigation in relational databases. *PVLDB*, 9(12):1017–1028, 2016.

[25] A. Kalinin, U. Cetintemel, and S. Zdonik. Interactive data exploration using semantic windows. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 505–516, New York, NY, USA, 2014. ACM.

[26] N. Kamat, P. Jayachandran, K. Tunga, and A. Nandi. Distributed and Interactive Cube Exploration. In *ICDE*, pages 472–483, 2014.

[27] S. R. Lay. *Convex Sets and Their Applications*. Dover Publications, 2007.

[28] H. Li, C.-Y. Chan, and D. Maier. Query from examples: An iterative, data-driven approach to query construction. *PVLDB*, 8(13):2158–2169, 2015.

[29] W. Liu, Y. Diao, and A. Liu. An analysis of query-agnostic sampling for interactive data exploration. Technical report, University of Massachusetts Amherst, 2016. Technical report UM-CS-2016-003.

[30] Y. Ma, R. Garnett, and J. G. Schneider. Σ-optimality for active learning on gaussian random fields. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 2751–2759, 2013.

[31] D. Mottin, M. Lissandrini, Y. Velegrakis, and T. Palpanas. Exemplar queries: Give me an example of what you need. *PVLDB*, 7(5):365–376, 2014.

[32] R. Neamtu, R. Ahsan, C. Lovering, C. Nguyen, E. A. Rundensteiner, and G. N. Sárközy. Interactive time series analytics powered by ONEX. In *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017, Chicago, IL, USA, May 14-19, 2017*, pages 1595–1598, 2017.

[33] G. Özsoyoglu and H. Wang. Example-Based Graphical Database Query Languages. *Computer*, 26(5):25–38, 1993.

[34] S. B. Roy, H. Wang, G. Das, U. Nambiar, and M. Mohania. Minimum-effort driven dynamic faceted search in structured databases. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM)*, pages 13–22, 2008.

[35] S. B. Roy, H. Wang, U. Nambiar, G. Das, and M. Mohania. Dynacet: Building dynamic faceted search systems over databases. In *International Conference on Data Engineering (ICDE)*, pages 1463–1466, 2009.

[36] B. Settles. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan Claypool Publishers, 2012.

[37] Y. Shen, K. Chakrabarti, S. Chaudhuri, B. Ding, and L. Novik. Discovering queries based on example tuples. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 493–504, New York, NY, USA, 2014. ACM.

[38] Sloan digital sky survey: Dr8 sample sql queries. `http://skyserver.sdss.org/dr8/en/help/docs/realquery.asp`.

[39] A. S. Szalay, P. Z. Kunszt, A. Thakar, J. Gray, D. R. Slutz, and R. J. Brunner. Designing and mining multi-terabyte astronomy archives: The sloan digital sky survey. In *SIGMOD Conference*, pages 451–462, 2000.

[40] B. Tang, K. Mouratidis, and M. L. Yiu. Determining the impact regions of competing options in preference space. In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, pages 805–820, New York, NY, USA, 2017. ACM.

[41] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, Mar. 2002.

[42] Q. T. Tran, C.-Y. Chan, and S. Parthasarathy. Query by output. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, SIGMOD '09, pages 535–548, New York, NY, USA, 2009. ACM.

[43] H. P. Vanchinathan, A. Marfurt, C. Robelin, D. Kossmann, and A. Krause. Discovering valuable items from massive data. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, pages 1195–1204, 2015.

[44] Z. Zhao, L. De Stefani, E. Zgraggen, C. Binnig, E. Upfal, and T. Kraska. Controlling false discoveries during interactive data exploration. In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, pages 527–540, New York, NY, USA, 2017. ACM.